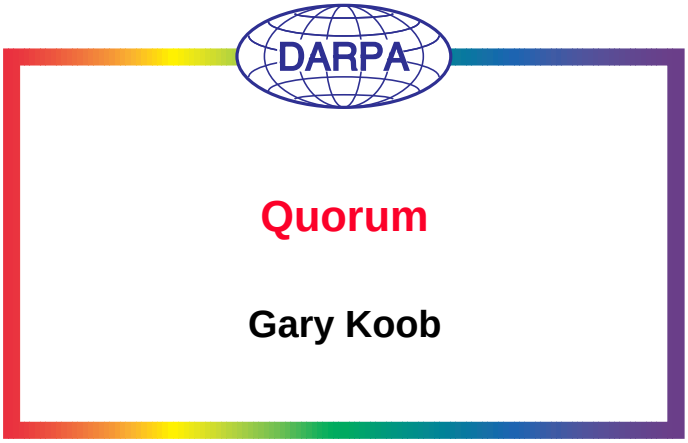
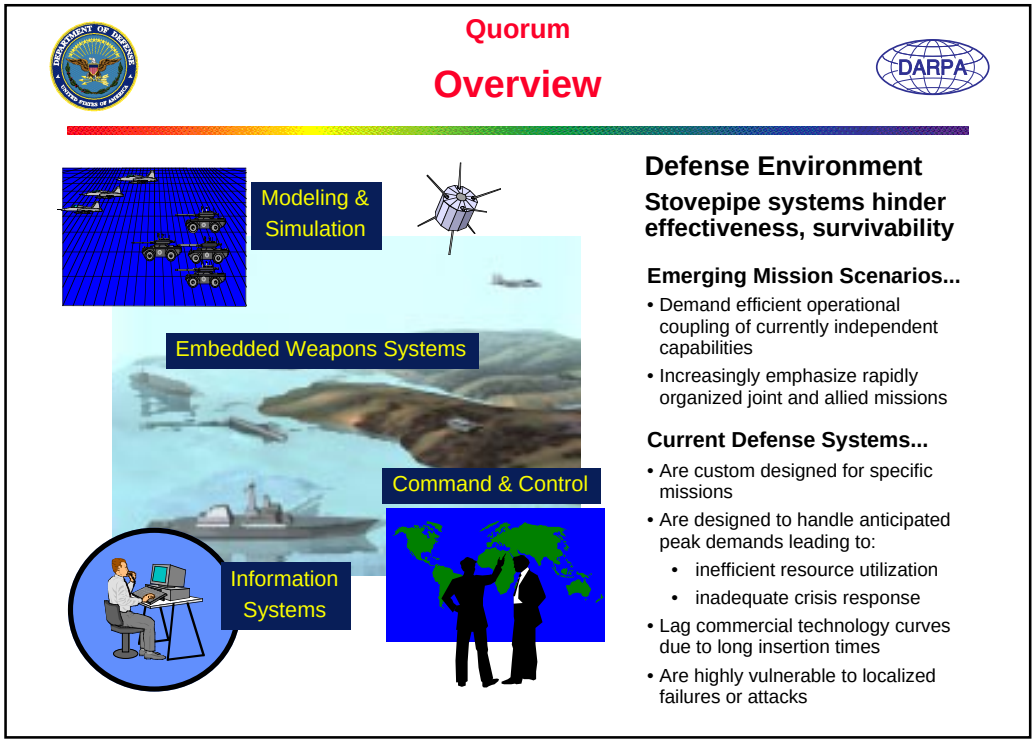


Quorum

Gary Koob

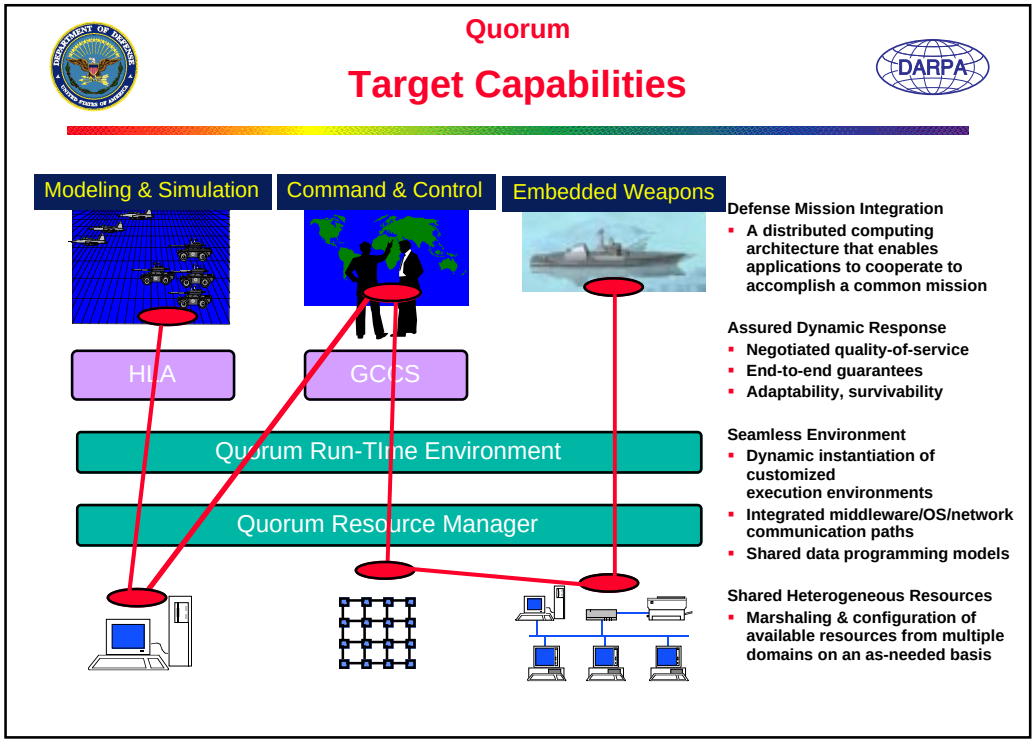


Emerging Defense mission scenarios are demanding tighter operational coupling of systems and capabilities that are currently independently developed and deployed. Simulation, for example, must be integrated with command and control to allow real-time course-of-action evaluation. Logistics and tactical systems must be similarly coupled to the global command and control system to ensure that all levels of the command hierarchy have rapid access to timely information and the ability to act decisively on it. Interoperability issues are further exacerbated by the increasing emphasis on rapidly deployable joint and allied task forces.

These scenarios are incompatible with the state of the practice. “Stovepipe” systems or subsystems are custom designed for specific functions with little attention paid to interoperability or the potential role that that subsystem might play in a mission. Platform capabilities are limited by the resources selected at design time and the tight binding of software to hardware severely limits the evolvability of the system but the high cost of redesign results in long-lived systems based on obsolete technology. Demands placed on the system, then, can easily exceed the capacity of the fixed resources allocated to it. Designing in excess capacity is inefficient and costly. The rigid assignment of functions to resources further leads to highly vulnerable systems.

The Quorum program envisions the equivalent of a global operating system (OS) as a solution to this problem. Such an operating system would dynamically discover and allocate global computing, communications, and information resources to meet the immediate demands of an application while preserving the illusion of a dedicated machine and the predictability of the stovepipe approach.

The graphic conceptualizes two users each executing his own application on a distinct virtual machine supported by resources assembled from a global pool.



This slide elaborates on the high-level vision presented on the previous slide.

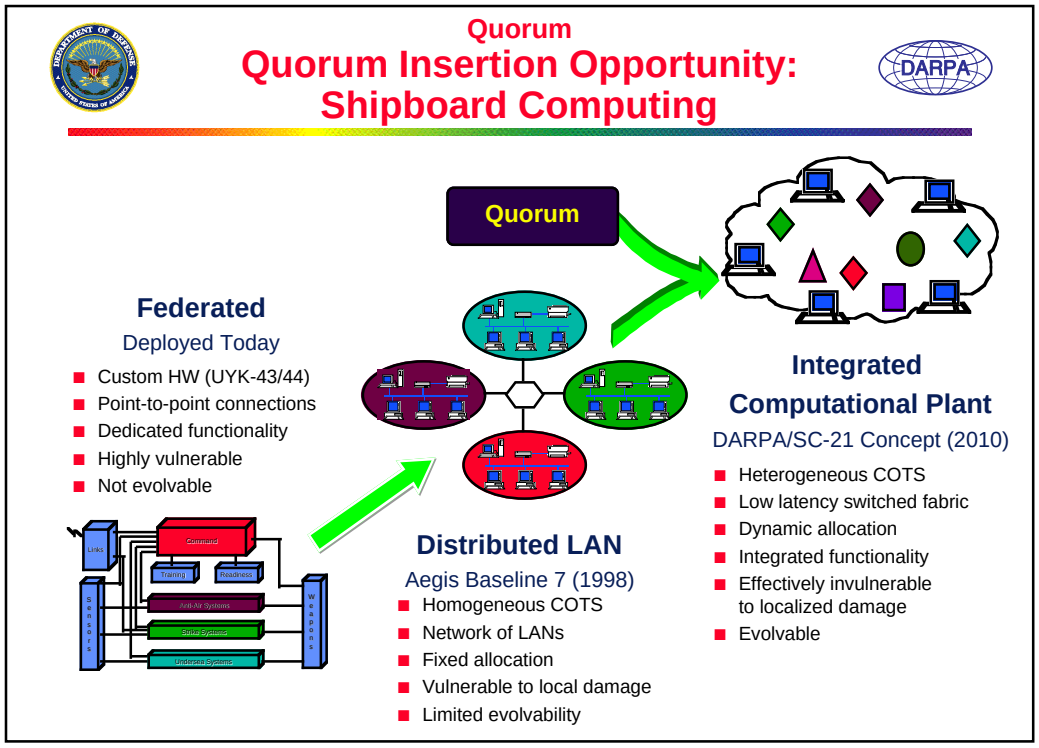
The Quorum resource manager would dynamically discover and allocate global computing, communications, and information resources to meet the immediate demands of an application. Capacity would no longer be limited by the resources available on a particular platform or at a particular site.

Once allocated these resources would be accessible to the application through Quorum Run-tIme Environment or virtual machine interface. The same application would execute correctly and efficiently regardless of the number, types, or locations of the constituent resources. The virtual machine need not be fixed but may be customized to the view appropriate to the application. The view required by a fluid dynamics simulation, for example, may be very different from that required by collaborative planning.

To be effective in a military context, this operating system would have to retain the compelling advantages of the dedicated stovepipe approach, e.g., optimization of software to resource capabilities, predictable end-to-end performance, and security, but in a shared, networked, highly dynamic environment, while adding further advantages of its own such as survivability. Achieving this over shared wide area networks is particularly challenging and will require innovative approaches to coupling the application directly to the network reducing the overhead introduced by middleware and operating system layers while preserving the protections, control, and flexibility supported by these layers.

The communications, memory, and execution interfaces of the baseline run-time environment form a substrate capable of supporting more application-specific environments such as the High-Level Architecture (HLA) for distributed simulation and the GCCS (Global Command and Control System) architecture.

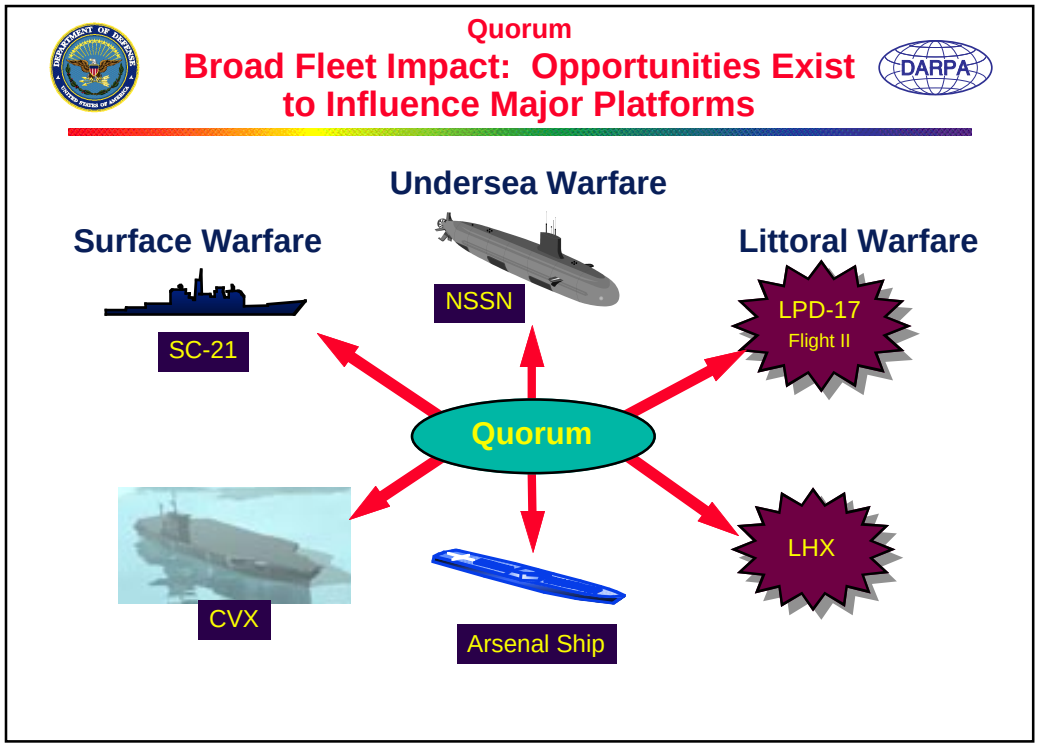
The common substrate and shared resource base will facilitate seamless integration of previously independent applications.



A compelling motivation and transition opportunity for Quorum is provided by the shipboard computing domain. This chart depicts the evolution of such systems, exemplified by the Aegis Combat Control System. The Federated architecture deployed on Aegis Baselines 1-6 is based on the stovepipe approach. The Distributed LAN architecture, developed through the DARPA/NAVY High Performance Distributed Demo (HiPerD) program, replaces the custom hardware with COTS processors, local area networks, and portable systems software emerging from DARPA technology programs. Each LAN, however, remains dedicated to a specific tactical function, e.g., Anti-Air Warfare. While fault tolerance is provided within each LAN, functions may not be reallocated across LAN boundaries. This architecture is scheduled for deployment in Aegis Baseline 7.

The Navy vision for the Aegis follow-on, the 21st Century Surface Combatant (SC-21) program, is that of an Integrated Computational Plant in which all shipboard computing functions, including administrative and mechanical, as well as tactical, share a ship-wide pool of resources, accessible through a low-latency network.

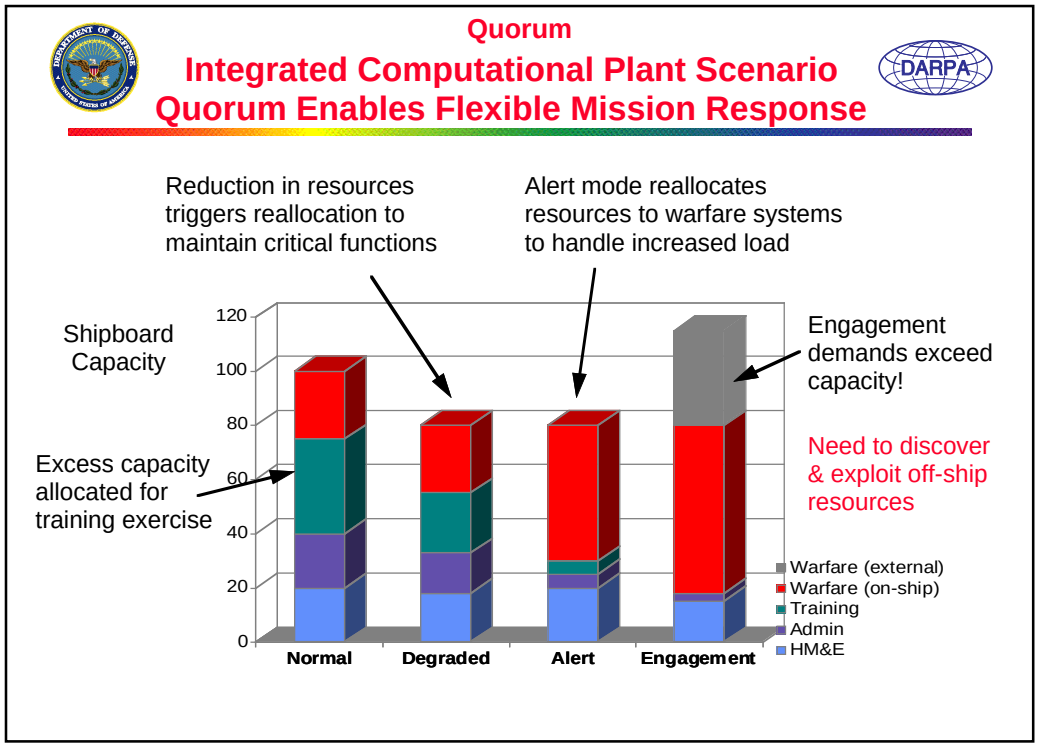
The similarities of this vision with the goals of the Quorum program suggest an ideal transition opportunity for Quorum technologies as well as a valuable Defense driver for Quorum projects.



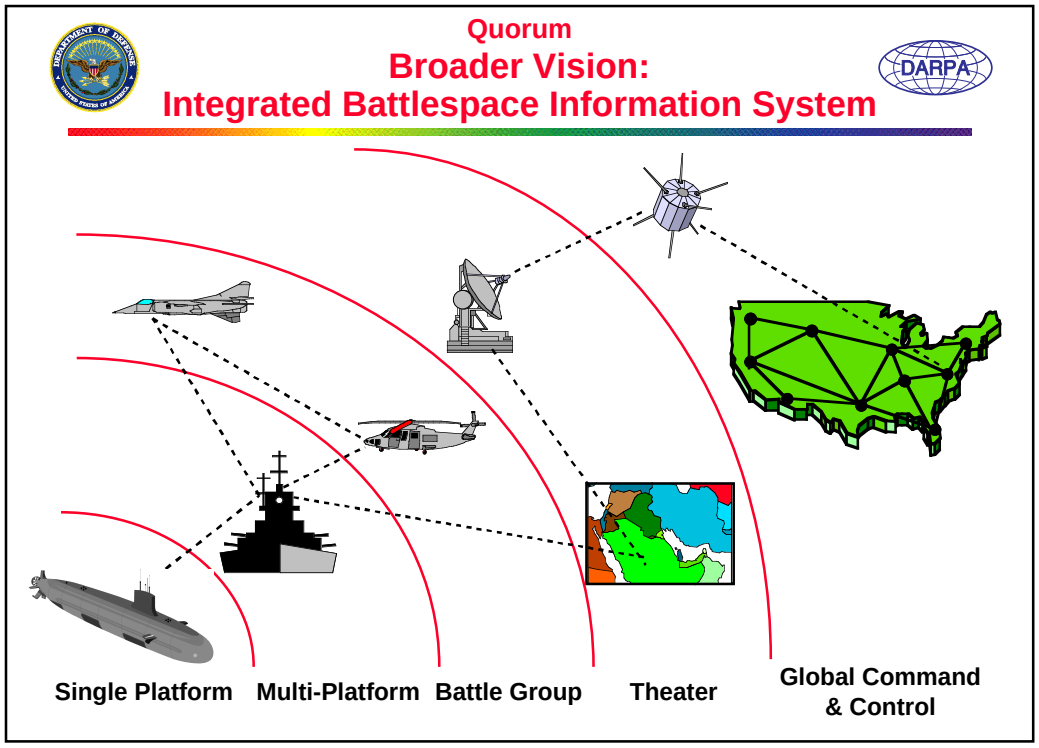
The Naval Surface Warfare Center, Dahlgren Division, which is leading the SC-21 project, has been chartered more broadly with investigating the concept of a standard architecture for all Naval platforms.

Programs that could be near- to mid-term adopters of such an architecture include not only SC-21, but the New Attack Submarine (NSSN), the DARPA/Navy Arsenal Ship Program, the next generation aircraft carrier (CVX), the LHX Amphibious Assault vessel, and Flight II of the LPD-17 amphibious support vessel.

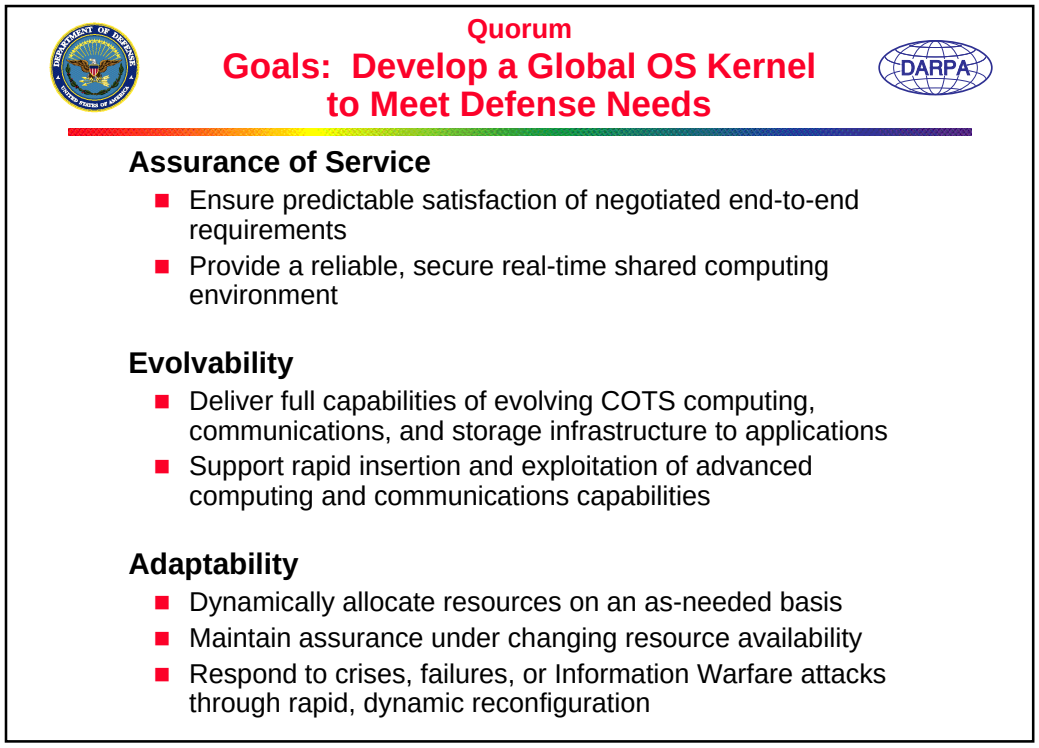
Through the SC-21 program, Quorum could have a major impact on the definition of a standard shipboard computing architecture and a broad spectrum of Fleet platforms and missions.



This chart shows conceptually how the available capacity of shipboard computing resources might be dynamically reallocated to various functions in different modes of operation. Excess capacity may be allocated to non-critical training exercises under low-demand conditions. In degraded modes, critical functions must be given priority over less critical ones and under crisis conditions the demands may exceed the available shipboard capacity. Thus, for crisis situations, it is desirable to have the ability to distribute functionality not just within a single platform, but among the (dynamically assembled) constituents of a battle group.

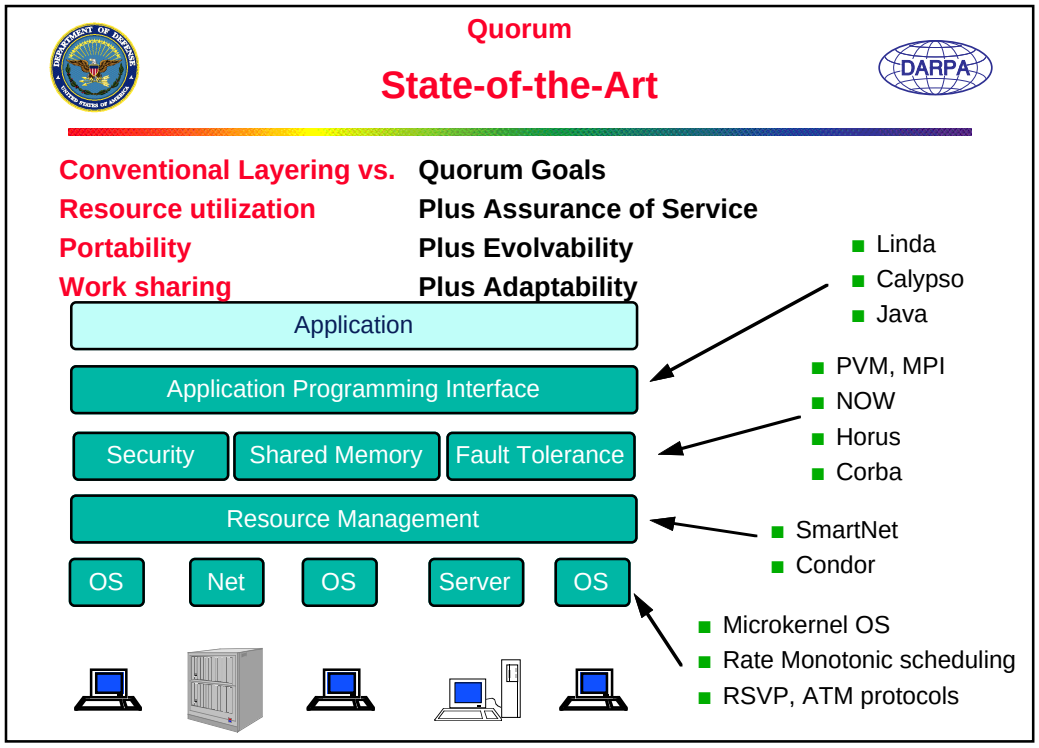


Quorum's vision therefore extends beyond the relatively controlled environment of a single platform to a computing environment capable of spanning a battle group and ultimately merging with the global command and control system.



The Quorum program is driven three high-level goals that represent a paradigm shift in the way distributed systems are conceived and constructed.

- First, the system must exhibit predictable behavior through negotiation and assurance of the quality of service visible to the application, i.e., end-to-end throughput, latency, jitter (timing variance), reliability, security, regardless of the processing, communications, and I/O resources allocated to the application or the aggregate load on the system. (Such assurance could be in the form of bounds or probabilistic guarantees reflecting residual nondeterminism or uncontrollable aspects of the system.)
- Second, the system must be evolvable, not only in the sense that the size and topology of the system should be allowed to dynamically change, but also in the sense that one should be able to rapidly insert new resources and automatically exploit their advanced capabilities, delivering their full power to applications. This implies components that augment traditional functional interfaces with descriptions of capabilities and enhanced control features (so that those capabilities may be exploited by other components.)
- Third, the system must be adaptable to changing system conditions, dynamically discovering and allocating available resources and reconfiguring in response to failures, priority demands, information warfare attacks, etc., while maintaining (or renegotiating, if necessary) the negotiated quality of service.

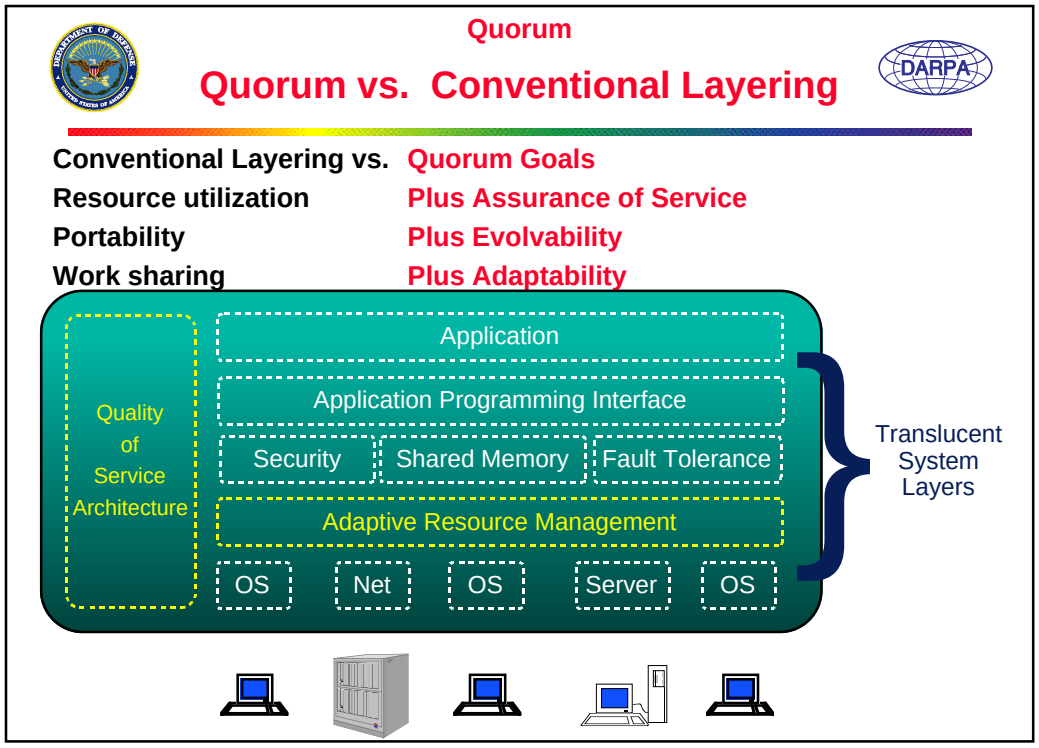


The traditional approach to distributed systems is to layer a distributed run-time environment and middleware services for security, communications, fault-tolerance, and resource management on top of native operating systems and the resources they control. As indicated on this slide, most current products and research prototypes target only one layer of this model, supporting a very limited distributed computing capability. The Linda System, for example, supports a language-based parallel programming model for a single application with no resource management or quality of service guarantees; PVM is a message passing library that supports heterogeneous computing at a low level. SmartNet supports coarse grain resource allocation but no run-time services. Technologies have also been developed at the OS and network layers for quality of service management, such as rate monotonic (real-time) scheduling and the RSVP network reservation protocol.

The two key points of this slide are:

- Various aspects of the distributed computing problem have been attacked over the past several years, providing a rich technology and experience base to build on. None of these approaches is completely satisfactory, however, in realizing the Quorum vision because their individual capabilities have not been integrated (and many capabilities are missing).
- (While the layering approach is effective in managing complexity through encapsulating functionality, this encapsulation hides critical implementation details and capabilities from higher layers making it impossible to achieve the assurance, evolvability, and adaptability goals of Quorum. Thus, even if these independent capabilities were layered on top of one another as suggested in the graphic, the system would at most exhibit the necessary but insufficient properties of efficient resource utilization, portability, and load balancing through work sharing.

The challenge of Quorum is to develop a new approach to structuring distributed systems that preserves the advantages of this layering approach while enabling the goals of assurance, evolvability, and adaptability to be achieved.



The Quorum approach is to achieve tighter coupling of the system layers through three key concepts:

- An overarching Quality-of-Service (QoS) architecture permits each layer to negotiate desired quality of service tradeoffs with lower layers and provide feedback on delivered quality of service to higher layers.
- Translucent system layers that augment conventional functional interfaces with systemic interfaces that allow client layers to access and control implementation and policy considerations relevant to the negotiated quality of service. The architecture defines the protocols that govern how these layers negotiate and convey relevant systemic information; the translucent layer component of the program develops the layers themselves with built-in flexibility for responding to negotiated QoS tradeoffs. The term “translucent” distinguishes this approach from the conventional principal of “transparency” where implementation decisions are completely hidden from the application but nevertheless manifest themselves through performance and reliability effects.
- The adaptive resource management component is essentially the coarsest level of the QoS hierarchy in that it is responsible for dynamically discovering and marshalling the available resources that can best satisfy the application demands and reconfiguring those resources as a last resort to maintain QoS assurance under changing system conditions. Finer-grain adaptations may be made within the translucent system layers.

Quorum

Objectives

Assurance of Service

- Demonstrate quality of service negotiation and adaptation to three different environments: LAN, WAN, and Internet
- Reduce end-to-end latency by order of magnitude to 125 us
- Demonstrate fault recovery in less than 100 milliseconds
- Demonstrate single real-time application spanning 3 trust domains

Evolvability

- Demonstrate ability to dynamically install and automatically exploit new resources with improved performance over time
- Demonstrate interoperability of advanced and legacy systems

Resource Management and Adaptability

- Scalable to 5,000 nodes; 10,000 schedulable entities; 100 domains
- Dynamic allocation in less than 3 seconds within 5% of optimal
- Demonstrate crisis mode response time of less than 30 seconds

This slide identifies the key quantitative objectives of the program. A few clarifications:

Under Assurance of Service …

- Ideally, the quality of service architecture should be able to adapt the system behavior to different environments without alteration of the application. Currently, assumptions about the capabilities of the environment are built into applications at design time resulting in applications that perform poorly when ported to a lower performance environment or that provide less information than feasible when ported to a higher performance environment. To demonstrate automatic adaptation, a single application will be executed without change in local area (LAN), advanced wide area (WAN), and commercial Internet environments.

Under Resource Management …

- The term “domains” refers to distinct trust domains or security enclaves. Allocating resources across such domains will require mutual authentication of the client, resource manager, and server as well as a means of providing a secure run-time environment that spans these domains controlling access to local resources and protecting the integrity of both the client task and the local server.

Quorum

Quorum Tasks

Quality-of-Service Architecture

- Protocols for negotiating and monitoring multi-criteria QoS
- Algorithms for mapping end-to-end requirements onto resources

Translucent System Layers

- Dynamically customizable OS and middleware services
- Integrated computation/communication services

Adaptive Resource Manager

- Resource models and discovery protocols
- Near optimal, adaptive multi-criteria allocation algorithms
- End-to-end scheduling algorithms

Integration and Demonstration

- Series of integrated reference Quorum prototypes
- Technology assessment against Defense requirements
- Demonstrations on key representative Defense problems

The four tasks of the Quorum program correspond to the concepts presented on the previous slides (QoS Architecture, Translucent layers, Adaptive Resource Manager) plus an Integration and Demonstration task that will evaluate and integrate component technologies, produce, distribute and support reference prototypes of a complete Quorum system to establish a broad user base (whose feedback will be valuable in improving the system), and evaluate and demonstrate the technology on key Defense problems, such as the 21st Century Surface Combatant problem.

Quorum

Quality-of-Service Architecture: Technical Innovations

QoS Specification

- Representations of alternative QoS regimes
  - Application perspective (end-to-end)
  - Capture complex tradeoffs in multi-dimensional space
  - Measurable
- Appropriate representations at other system levels

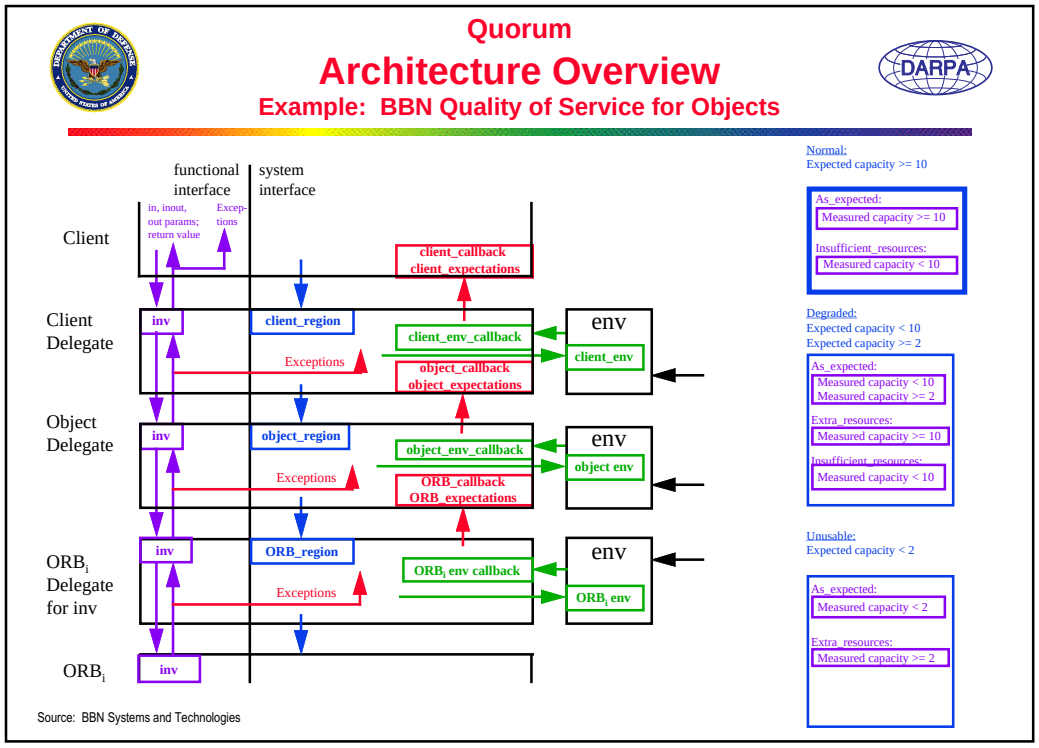
QoS Negotiation

- Protocols for reconciling available QoS with demands
- Algorithms for propagating constraints through system layers

QoS Assurance

- Instrumentation for dynamic monitoring of delivered QoS
- Algorithms for characterizing, predicting delivered QoS
- QoS maintenance through dynamic feedback, adaptation

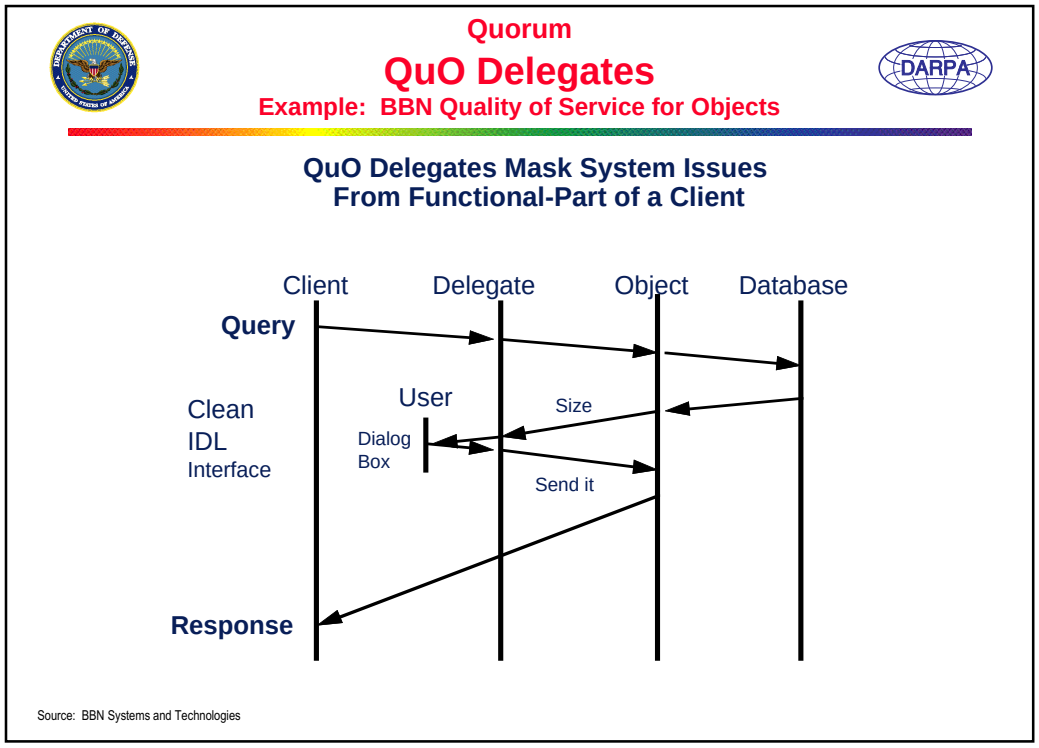
This slide outlines the key technical components and anticipated innovations of the Quality of Service task.



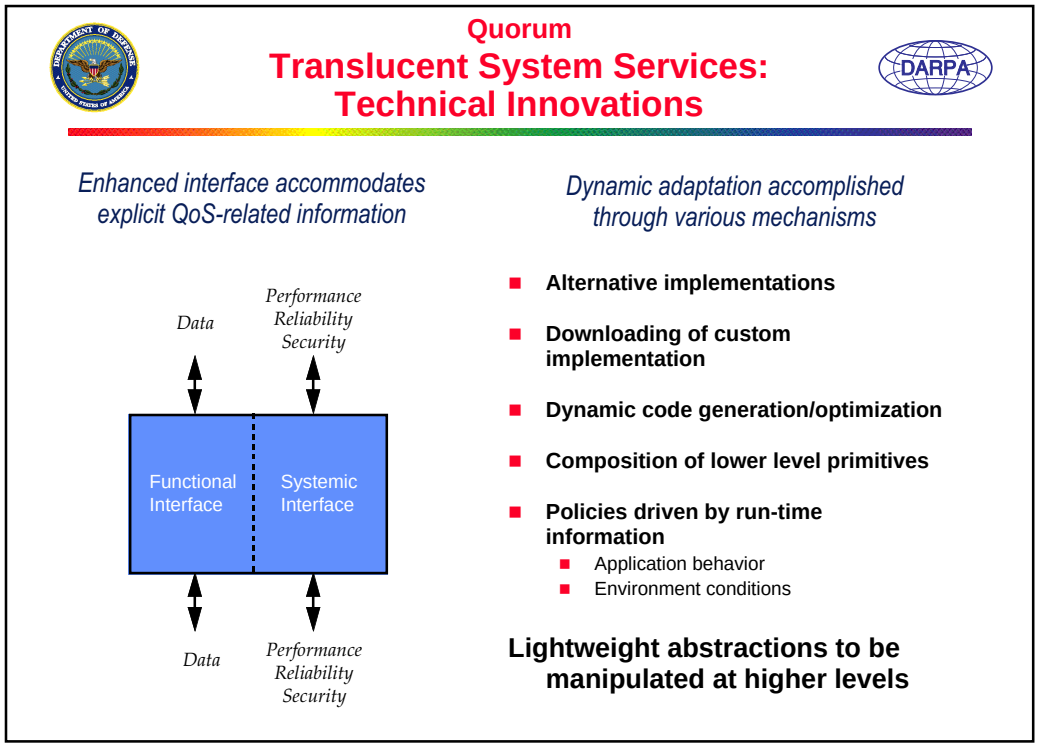
An example of the QoS architecture task is provided by the Quality of Service for Objects (QuO) project at BBN Systems and Technologies. QuO is based on the CORBA architecture for shared objects but augments that architecture to accommodate assured QoS over wide area networks. First, the CORBA functional interfaces are augmented by systemic interfaces supporting QoS negotiation and maintenance of system state information. Control of the CORBA clients, objects, and object request brokers (ORBs) is effected through delegates, or proxies, that reside in the client's address space. Operation of the system at each level is partitioned into mutually exclusive QoS regions defined by constraints on various system parameters, such as bandwidth, latency, capacity, etc. Each region of operation may require a distinct implementation of the service provided at each level. For example, under low bandwidth conditions, it may be necessary to employ data compression. The figure at the far right shows how these regions are specified.

The right side of the architecture diagram illustrates the systemic interface which allows querying and negotiation of QoS regions of lower layers as well as monitoring of critical system conditions that may trigger a transition to a new QoS region. Up calls permit notification of changes to higher levels which may turn trigger appropriate adaptation at those levels.

The architecture also accommodates a unified exception handling mechanism permitting adaptation to run-time exceptions, such as software and hardware faults, as well as to general performance conditions monitored through the environment variables.



This slide simply illustrates schematically how the BBN delegate architecture separates the functional aspects of the system (which the client cares about) from the automatic QoS management activities. The latter may occasionally require explicit guidance or control from the user, but this does not interfere with the information flow through the application.



The QoS Architecture allows layers to negotiate and monitor quality-of-service parameters through augmented interfaces such as that shown on the left and depicted in the BBN example. The challenge for the Translucent System Layers task is to populate this architecture with services that exhibit the flexibility to respond to negotiated QoS constraints by altering or customizing their implementation or behavior in some way.

The list on the right identifies some of the innovative concepts being explored under the program to effect this customization. The simplest concept is to provide alternative implementations of a service, each optimized to a particular QoS region. As the definition of these regions is often application-specific, this approach is inherently limited.

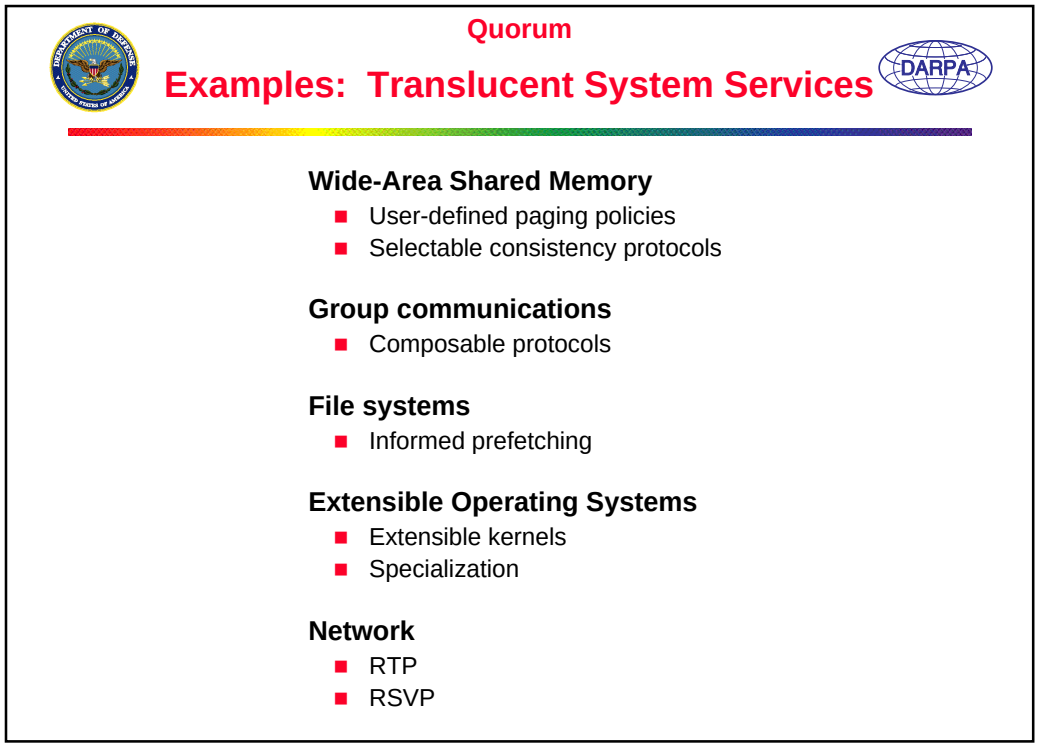
An extreme variation of this is to allow applications to download their own custom implementations. This is being explored in the operating systems area along with all of the safety and security implications of the approach.

Other approaches are examining the use of compiler technology to dynamically optimize an existing implementation to the context in which it will be used or to dynamically generate code from a specification of requirements.

A different class of approach is to structure the service as a collection of primitive services each supporting a distinct increment of functionality. These primitives may then be dynamically composed to provide exactly the high-level service required, avoiding the overhead of superfluous functionality.

A less intrusive approach is to separate out policy issues from mechanisms, allowing efficient mechanisms to be driven by externally specified policies or run-time data.

Finally, the service can export a very lightweight abstraction that provides minimal functionality, allowing heavier application-specific abstractions and policies to be built on top of it.

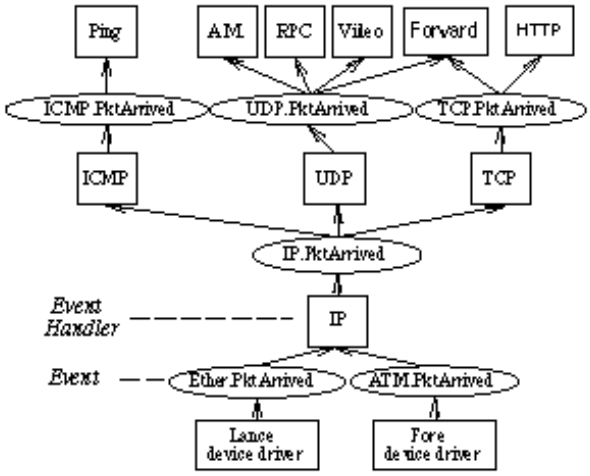


This slide simply outlines some of the essential system layers to which the translucent principles must be applied to realize the Quorum vision.

Quorum

Example: SPIN Packet Filter

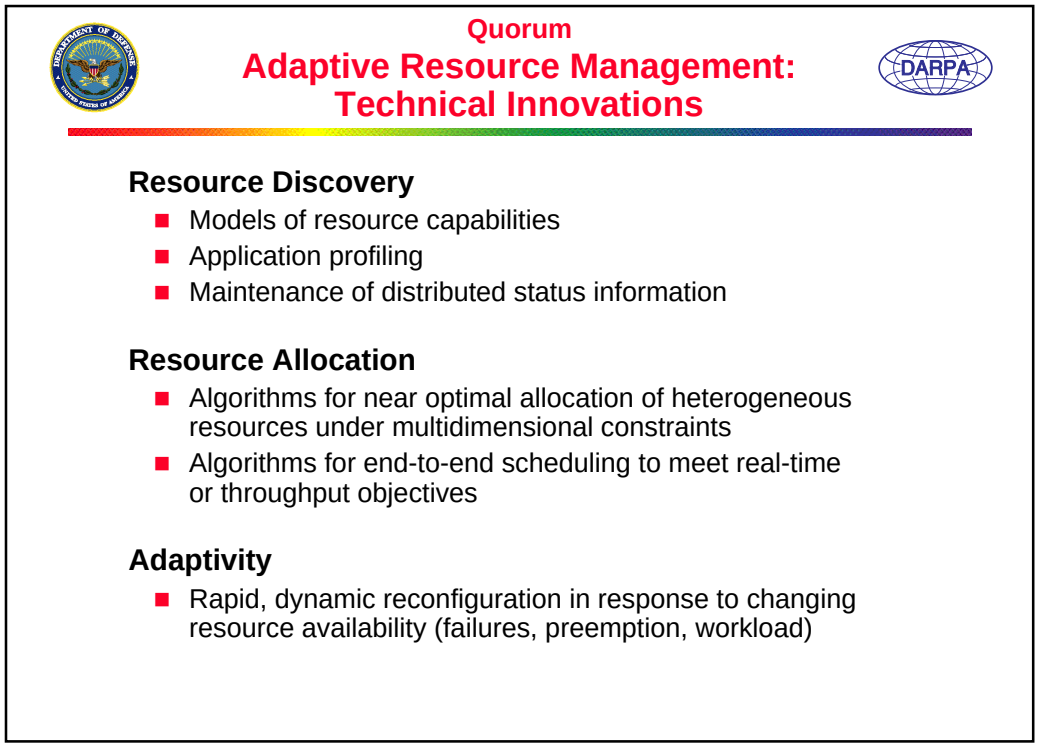
Extensible kernel enables application-specific packet processing



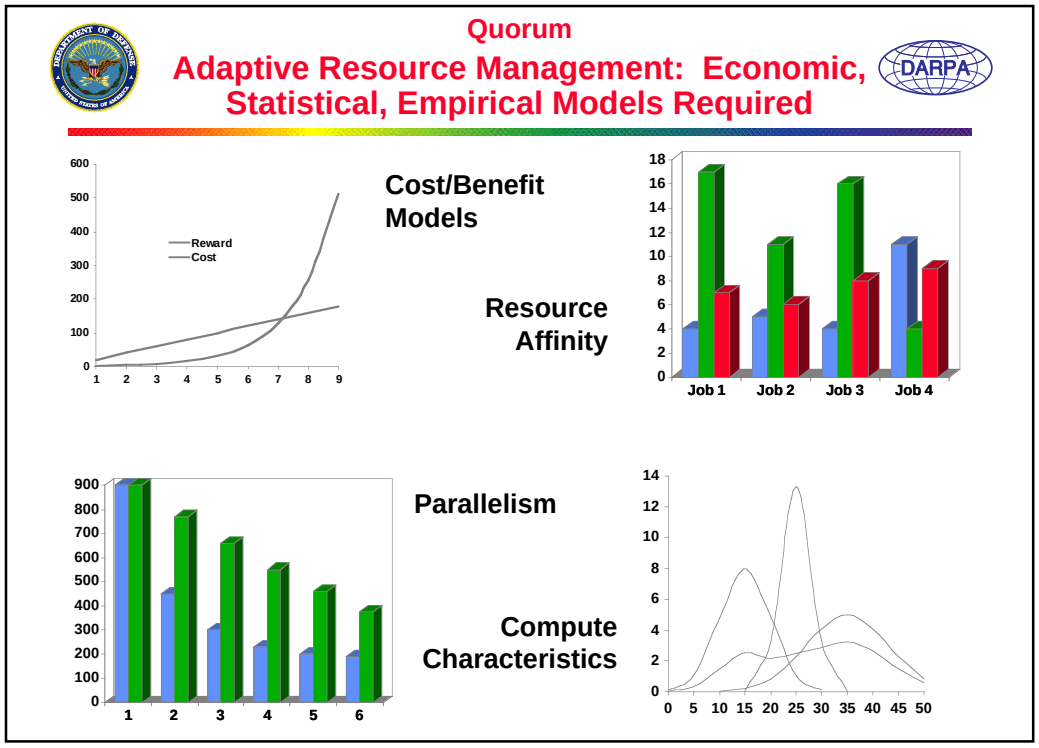
This is an example of how the SPIN operating system (U. Washington) employs extensible kernel technology to achieve high-performance communications by customizing packet processing to application requirements.

The figure shows a protocol stack that routes incoming network packets to application-specific endpoints within the kernel. Ovals represent events raised to route control to handlers, which are represented by boxes. Handlers implement the protocol corresponding to their label.

This exemplifies a combination of the composition approach to customization with downloadable modules.



The third task is adaptive resource management, which requires technical innovations in the areas of resource discovery, resource allocation algorithms and adaptivity.



This slide conceptually identifies some of the key dimensions of the resource allocation problem.

Some of the most promising ideas involve the use of economic models for resource allocation in which prices are assigned to resources and clients are allocated budgets. The chart in the upper left illustrates how cost/benefit tradeoffs must be assessed by the clients in determining how best to spend their budgets to achieve acceptable service.

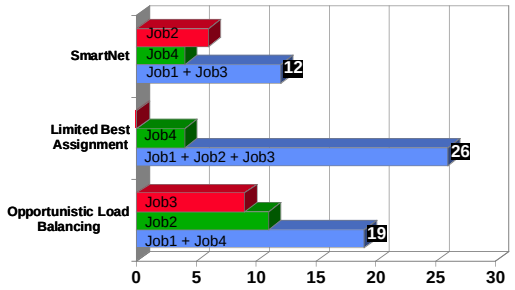
In heterogeneous systems, performance is highly dependent on “affinity”, the degree to which a particular resource efficiently supports the computation. The chart in the upper right plots execution time for four different jobs on three different machine architectures, indicating that Job 4 performs best on the second machine, while Jobs 1-3 all perform best on the first machine.

The chart in the lower left shows speed-up curves for a parallel job executing on two different clusters of workstations. These curves must be taken into account in allocating parallel jobs.

Achieving predictability of performance requires not only assessing performance on particular resources but understanding the variation in that performance as a function of the data set. The graph in the lower right shows that the execution time of a particular application may be essentially unpredictable, having an essentially flat distribution (purple curve). Profiling of the application can be used to decompose this distribution into three sharper conditional distributions, each valid on a particular subdomain, enabling much greater predictability of the execution time. This set of distributions constitutes the compute characteristic of the application.

Quorum

Example: Maximizing Throughput of Independent Jobs



Best strategy exploits both affinity and load information

Completion Time of Last Job (normalized to super-optimal)

	Scalable Architecture	Architecture Mix	Architecture Mix
Jobs/Machines	500/100	500/100	1000/500
LBA	26.5	27.5	105.5
OLB	1.45	1.28	1.83
SmartNet	1.13	1.06	1.29

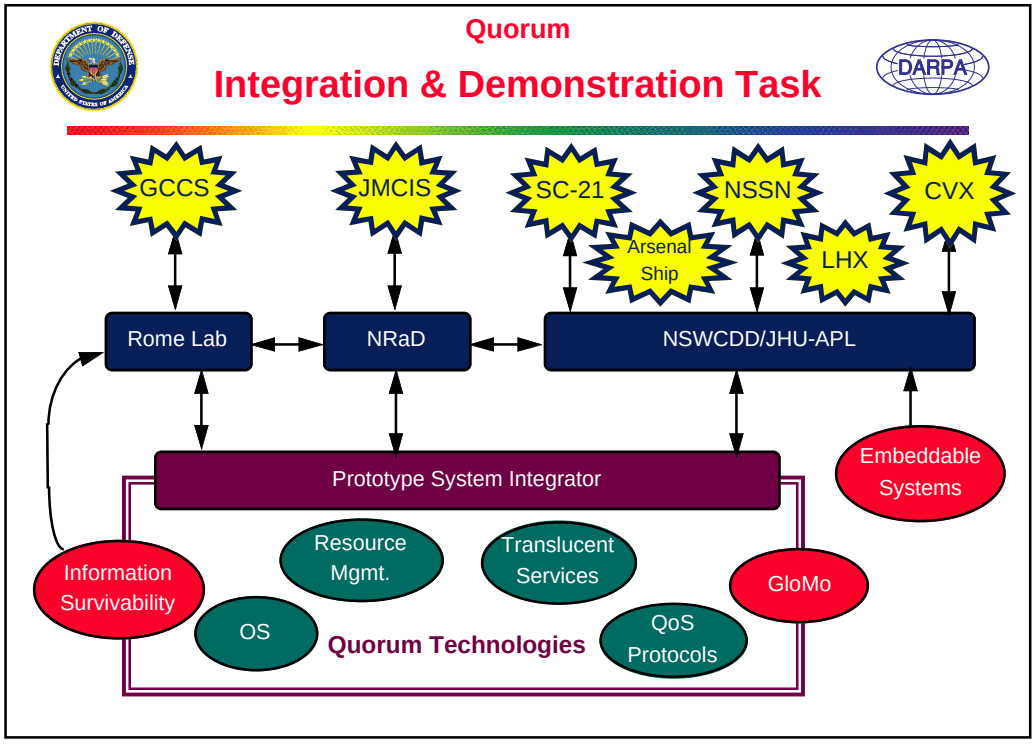
Source: Richard Freund, NCCOSC

This chart was adapted from data provided by Richard Freund at the Naval Command Control and Ocean Surveillance Center (NCCOSC).

The bar chart compares three strategies for resource allocation. Opportunistic Load Balancing only attempts to balance the load on the system. Limited Best Assignment takes account only of affinity information in assigning jobs to resources. Freund's approach, called SmartNet, combines the criteria and significantly outperforms the other strategies, as measured by the completion time of the last job.

The table confirms this result, presenting results for three sets of experiments varying the number of machines, their types and the numbers of jobs scheduled. The results are normalized to "superoptimal", i.e., an analytic lower bound on the best schedule. (Determining the actual optimal allocation is an intractable problem for large numbers of jobs and tasks.)

The table indicates that the SmartNet approach performs within 6% of optimal on the middle experiment.



This slide presents an overview of Quorum's Integration and Demonstration Task.

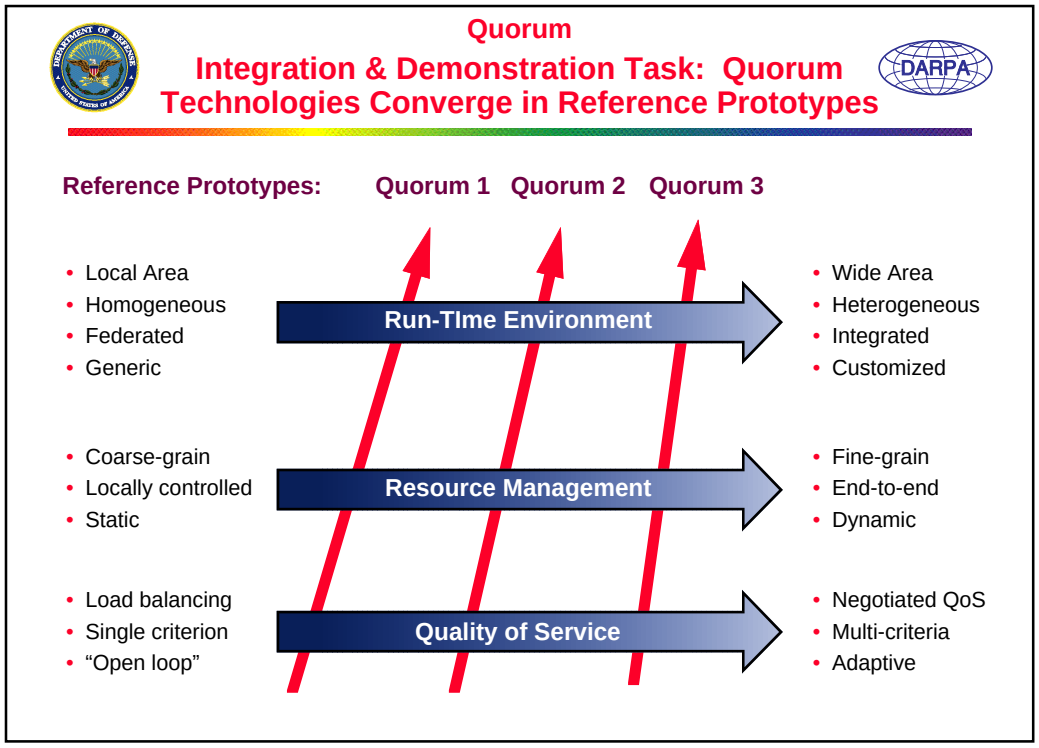
The strategy calls for a prototype system integrator to integrate and evaluate the constituent technologies developed under the other three tasks along with relevant technologies developed under the DARPA Information Survivability and Global Mobile Computing (GloMo) programs. The products of this effort will be reference prototypes and integrated technologies that will be delivered to DoD activities for evaluation in the context of military systems under development.

The three key receptors will be the Naval Surface Warfare Center, Dahlgren Division (and their prime contractor, Johns Hopkins University Applied Physics Lab) and the two principal agents for the program, NRaD, and Rome Lab.

NSWC will incorporate Quorum technologies into a testbed for the SC-21 program. As NSWC will be exploring concepts and technologies for a general shipboard computing architecture, transition paths out of this testbed include not only SC-21 but other major platforms as well: NSSN (New Attack Submarine), CVX (aircraft carrier), Arsenal Ship, and LHX (amphibious assault vehicle). As SC-21 also has requirements for high performance signal processing, it will also be evaluating relevant technologies being developed under DARPA's Embeddable Systems program.

NRaD will explore insertion of Quorum technologies into the Navy's Joint Maritime Command Information System (JMCIS) and will work with NSWC to explore using Quorum technology to more effectively link the JMCIS command and control environment to the SC-21 combat control environment.

Finally, NRaD and Rome Lab will identify transition opportunities for Quorum technologies into the Global Command and Control System (GCCS).



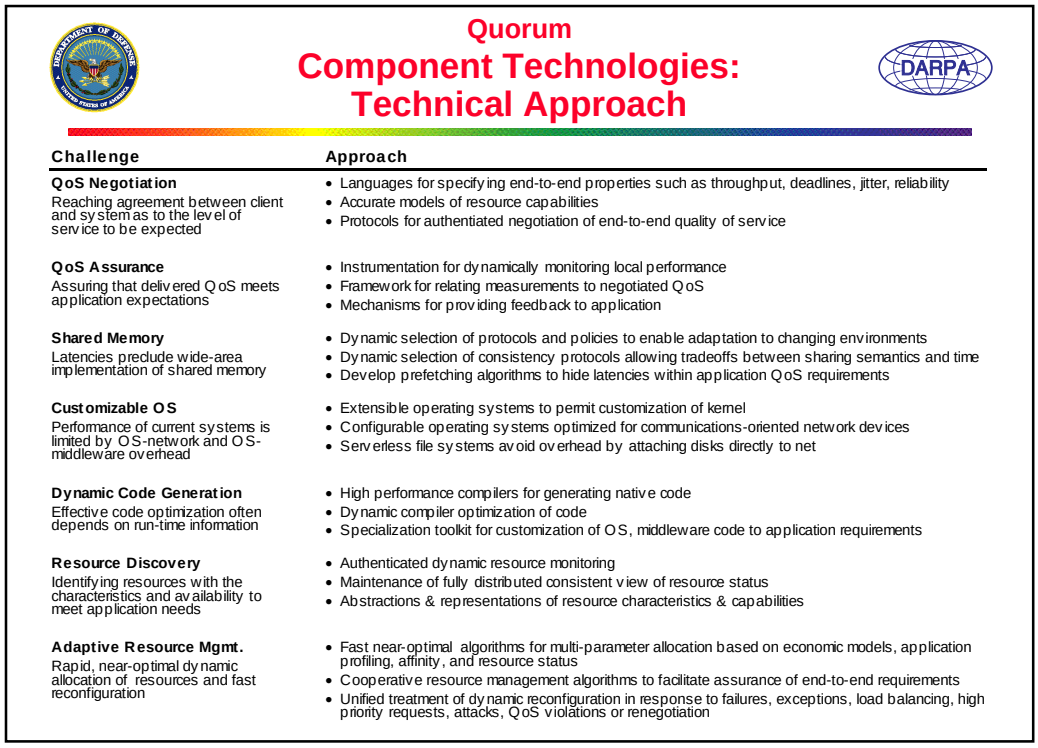
Quorum's three technology tasks will be developing technologies that advance from the current state of the art properties listed on the left to the desired properties listed on the right. Progress in these advances will proceed at different rates, as suggested by the red arrows, but snapshots will be taken at three key points in the program, producing a series of three complete reference prototypes of successively greater capability (see next slide).

Quorum

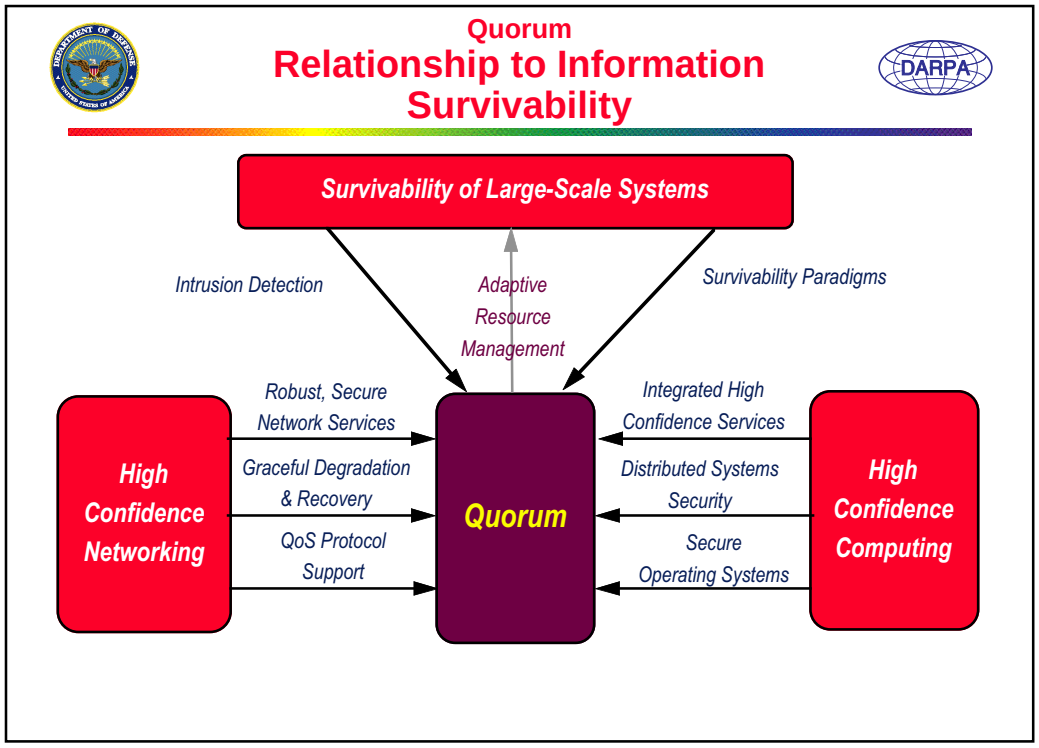
Quorum Reference Prototypes

	Quorum 1	Quorum 2	Quorum 3
Delivery	FY1998	FY1999	FY2001
Scalability	200 nodes	1000 nodes	5000 nodes
QoS Criteria	End-to-end performance	End-to-end soft real-time	End-to-end hard real-time
Allocation Optimality	Within 30%	Within 15%	WIthin 5%
Negotiability	Admission control	Negotiable QoS	Adaptive
Security	Authentication	End-system security	End-to-end security
Fault Tolerance	High availability	Time-constrained; negotiable	Adaptive

This slide presents strawman descriptions of the three major prototypes to be developed under the program, outlining the anticipated progress in several key parameters to be demonstrated by the prototypes.



This slide summarizes some of the key technical challenges to be addressed in the program and the approaches to be taken in addressing them. This slide covers only the core technology tasks -- not the Integration and Demonstration task.

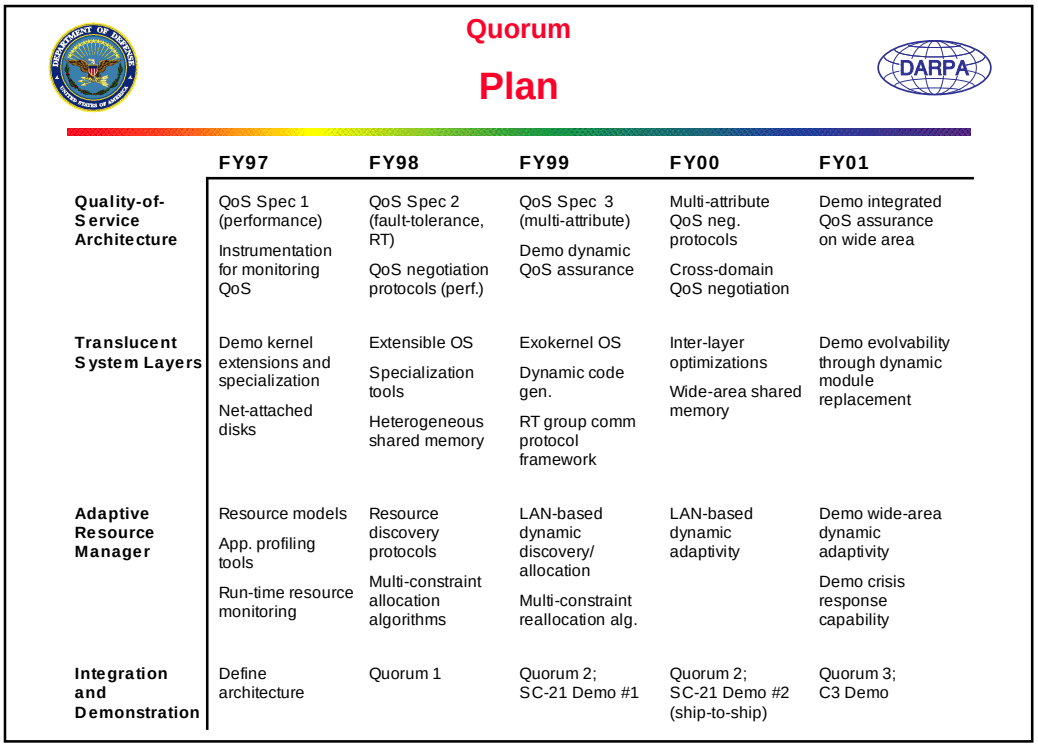


The Quorum Program is strongly coupled to the Information Survivability (IS) program. In general, the IS program is developing modular core technologies supporting security, fault-tolerance, real-time assurance, and survivability. Quorum will integrate these technologies into a general distributed computing system architecture.

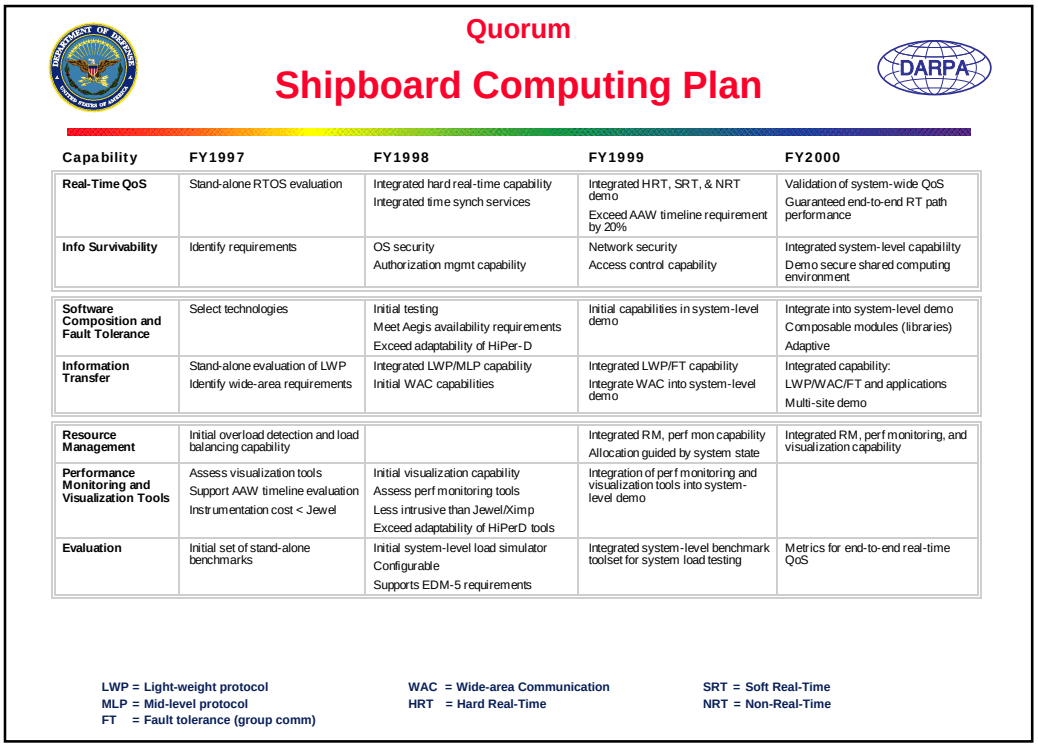
The IS High Confidence Networking task is developing technologies to provide robust, secure network services (such as name services and routing), graceful degradation and recovery at the network level, and protocols for supporting negotiated QoS at the network level (Quorum is primarily concerned with exploiting network capabilities in a system and application context--not with developing new network services).

The IS High Confidence Computing task is developing fundamental technologies for secure operating systems, secure distributed computing, and protocols for such integrated middleware services as real-time group communications for fault tolerance.

The Large-Scale Systems task is developing new paradigms for survivability that could be layered on top of Quorum and intrusion detection approaches that will be integrated into Quorum's QoS framework. Quorum, in turn, will be developing the adaptive resource management approaches necessary to support survivability goals.



This is the Quorum program plan identifying key developments and their target completion dates.



This plan, provided by NSWC, outlines the schedule of the SC-21 program, identifying the key technology developments required in each year. SC-21 will draw on technologies developed under Quorum to meet these goals.